ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Gocker e Node al DISI

Fabio Vitali + Francesco Sovrano

Corsi di laurea in Informatica e Informatica per il Management

Alma Mater – Università di Bologna

Node e npm

Creatura di una persona sola, Ryan Dahl (2009), per fornire bi-direzionalità alle applicazioni web (tecnologia *push*).

Un'applicazione nativa server-side, che permette di fare chiamate I/O guidate da eventi, non-bloccanti.

Per uniformare la realizzazione di applicazioni miste server/client, decide di usare un motore Javascript anche server-side

- Non è la prima volta, già Microsoft con ASP usa Javascript server-side
- Utilizza un interprete di Javascript molto potente, V8 di Google

Ad esso viene aggiunto un package manager, ***npm***, che permette di installare velocemente librerie e pacchetti associati.

Usare node sulle macchine del DISI

- Il progetto di TW richiede l'uso di node come motore dell'applicazione server-side.
- Inoltre le regole del corso impongono l'uso di una macchina DISI per ogni servizio server-side del progetto.
  - Ogni struttura dati e ogni servizio sw creato da studenti UniBo deve essere fornito da una macchina *.cs.unibo.it
- Il DISI mette a disposizione DUE modi per attivare un servizio node:
  - attraverso *gocker*, un servizio docker di virtualizzazione di una macchina online (*container*)
  - Via linea di comando su una porta alta (>1024)
- Entrambi richiedono qualche competenza di linea di comando (poca roba)
- Entrambi funzionano sullo spazio dati e con i permessi dello studente, e non dell'amministratore o di root.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Uso della shell Unix per chi non ricordasse Sistemi Operativi

Un minimo di sintassi shell (1/3)

Alla shell delle macchine si accede via ssh:

```
> ssh eva.cs.unibo.it
fabio@eva.cs.unibo.it's password:
```

O equivalentemente:

```
> ssh fabio.vitali@eva.cs.unibo.it
fabio.vitali@eva.cs.unibo.it's password:
```

Appare il prompt della shell, che varia da caso a caso:

```
nomeutente@nomemacchina:dir$
fabio@eva:/home/web/siteXXXX$
```

Un minimo di sintassi shell (2/3)

- Per cambiare directory:
  - cd /path/assoluto `fabio@eva:~$ cd /home/web/siteXXXX`
  - cd nomelocale `fabio@eva:/home/web/siteXXXX$ cd html`
- Per mostrare il contenuto della directory:
  - ls (lista semplice dei file visibili della directory corrente)
  - ls -l (lista completa dei file visibili della directory corrente)
  - ls –al (lista completa dei file visibili e invisibili della directory corrente)
  - `fabio@eva:/home/web/siteXXXX$ ls -al`

```
total 308
drwxr-xr-x    7 root     root       4096 May  4 09:26 .
drwxr-xr-x 3082 root     root     286720 May  4 15:48 ..
drwxrwsr-x    2 site1833 site1833   4096 May  4 09:26 cgi-bin
drwxrwsr-x    2 site1833 site1833   4096 May  4 09:26 data
drwxrwsr-x    3 site1833 site1833   4096 May  8 16:31 html
drwxr-x---    2 webadm   site1833   4096 May  4 09:26 log
drwxrwsr-x    2 site1833 site1833   4096 May  4 09:26 scripts
```

Un minimo di sintassi shell (3/3)

- Per cambiare i permessi di accesso:
  - chmod permessotesto nomefile `chmod a+x file.php`
  - chmod permessobinario nomefile `chmod 755 file.php`
- Per modificare un file di testo:
  - nano nomefile o pico nomefile
  - vi nomefile oppure vim nomefile
  - emacs nomefile
  - Sono tutti editor molto diversi tra di loro. *Io uso vi* .
- Per avere istruzioni su un comando Linux
  - man comando `man ls`
  - man –k keyword `man -k files`

Imparate la shell!
Imparate ad usare i vostri strumenti

Macchine GNU/Linux

- Le macchine hanno nomi di personaggi di opere.
- Alcuni esempi:
  - annina.cs.unibo.it
  - berta
  - ceprano
  - dorina
  - eufemia
  - frazier
  - giovanna
  - …
- L'elenco completo è su: http://www.informatica.unibo.it/it/servizi-informatici/accesso-da-remoto

Il file system condiviso del DISI

Tutte le macchine Linux del dipartimento hanno una struttura del file system simile, e condividono alcune directory:

- /home
- /public

Questo significa che da qualunque macchina vi colleghiate, potrete sempre accedere alla stessa directory.

Dentro ad /home si trovano le home directory di utenti e progetti e siti web, come segue:

- /home/faculty, /home/guest-fac, /home/phd-students, /home/postdoc, /home/staff, /home/students: home directory di specifici utenti umani, a seconda del ruolo e della durata del diritto d'accesso.
- /home/esami, /home/projects: home directory di progetti di ricerca e strutture temporanee di vario tipo, attivate e disattivate secondo necessità
- /home/nws, /home/web: home directory di siti web, le uniche su cui sia attivato un server http

La directory /home/web/siteXXXX/

- Contiene cinque directory:
  - cgi-bin
  - data
  - html
  - log
  - scripts
- Di queste, ci serve SOLO html, che è il posto dove mettere script e file del nostro sito web.
- Ad /home/web/siteXXXX/html/ corrisponde la directory di base di http://siteXXXX.web.cs.unibo.it/ e di http://siteXXXX.tw.cs.unibo.it/ .

Node come motore HTTP

Node può elegantemente sostituire PHP, Python, Perl e ogni altro approccio server-side tradizionale. Di seguito un esempio di un'applicazione che restituisce un frammento HTML:

```
var http = require('http');

http.createServer(
   function (request, response) {
      response.writeHead(200, {'Content-Type': 'text/html'});
      response.end('<h1>Hello World</h1>');
   }
).listen(8000);
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Node e npm su Unix

Docker

Node da linea di comando

```
var http = require('http');

http.createServer(
   function (request, response) { ... }
).listen(8000);
```

- Il modo più semplice per attivare node è da linea di comando:
  - `fabio@eva:~$  cd /home/web/siteXXXX/html/`
  - `fabio@eva:/home/web/siteXXXX/html/$ node index.js`
- A questo punto node risponde all'indirizzo:
  - `http://eva.cs.unibo.it:8000/`

PATH e shell

- Può succedere che otteniate errori tipo

```
fabio.vitali@eva:~$ node index.js
-bash: node: command not found
fabio.vitali@eva:~$ npm install express
-bash: npm: command not found
```

- Questo significa che non avete la directory di node e npm nel PATH (che sono le directory cercate automaticamente dalla shell).
  - E' necessario o specificare il path assoluto di node e npm...
    - `/usr/local/node/bin/node index.js`
    - `/usr/local/node/bin/npm install express`
  - oppure inserire la directory di node nel PATH:
    - `cd $home`
    - `nano .bash_profile`
    - Aggiungete la seguente riga: export PATH=/usr/local/node/bin/:$PATH
    - Riavviate la shell, lanciando il comando: exit

Limiti e problemi di node da linea di comando

- Funziona solo finché la shell è aperta.
  - Quindi non possiamo scollegarci finché vogliamo che funzioni
  - Non è vero, c'è un modo, ma non è il momento per parlarne.
- Funziona solo dalla macchina da cui abbiamo lanciato l'eseguibile.
  - Quindi l'URI deve specificare esattamente la macchina scelta.
- Funziona solo su porte alte > 1024, e la porta deve essere libera per quella macchina.
  - quindi l'URL da usare DEVE specificare la porta
  - Suggerimento: poiché Gocker funziona necessariamente sulla porta 8000, usate la porta 8000.
- Ci può essere solo UN servizio attivo su una data porta e un dato server
  - Quindi se usate tutti la porta 8000 dovete tutti usare server diversi.
- Utile nelle fasi di programmazione e debug.
- Una volta che il server funzioni dovrete cercare qualcosa di più stabile.

Docker (1/3)

- E' un fornitore di container software, ovvero macchine virtuali minimali in grado di eseguire una sola applicazione senza richiedere tutta l'infrastruttura di un sistema operativo.
- Il DISI utilizza docker per fornire macchine virtuali con un software preinstallato:
  - node
  - python
  - php
  - mongoDB
- **Sebbene in teoria sia possibile, questo laboratorio non accetta immagini con sw preinstallato diverso da questi.**
- I container docker accedono al file system condiviso dei server del dipartimento, quindi possono essere attivati su qualunque directory del file system. Noi tradizionalmente usiamo /home/web/siteXXXX/html/

Docker (2/3)

- La nostra implementazione di docker risponde via ssh all'indirizzo gocker.cs.unibo.it. E' possibile accedere via ssh a gocker solo da una macchina di laboratorio. Si deve dunque accedere via ssh prima ad una macchina di laboratorio (es. eva.cs.unibo.it) e da quella macchina accedere sia ssh a gocker.cs.unibo.it.
  - I siti web studenti per ragioni storiche utilizzavano una specifica macchina virtuale chiamata Golem. ***Gocker = Golem + Docker***.
  - Oggi Golem non esiste più o esiste per altri scopi, ma il nome è rimasto.
- Alla partenza Gocker fornisce una shell MOLTO minimale, senza navigazione sul file system, con pochi comandi:
  - create: crea un container con un solo servizio (node o php o python)
  - list: fornisce un elenco di tutte le directory dell'utente su cui è possibile o è stato attivato un container
  - remove: rimuovi un container
  - restart: ferma e riavvia un container
  - exit: esci dalla shell di docker
  - help: mostra un semplice aiuto sui comandi

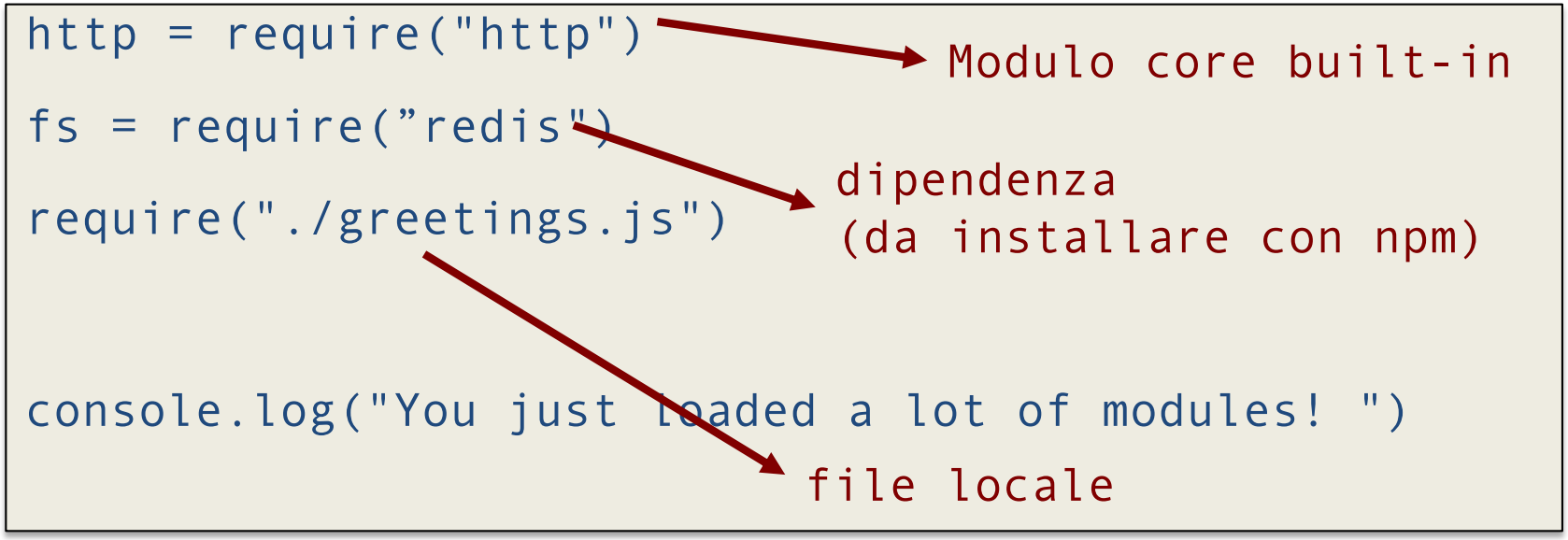
Docker (3/3)

Il comando più interessante è create:

- `create technology site [script]`
- ad es.: `create node siteXXXX index.js`
  `create py34-wsgi siteXXXX`
- technology è uno di vari come:
  - `static`,
  - `node`,
  - `php7`, `php5`,
  - `py34-wsgi`,
  - `etc.`
- sitename è il dominio di V livello associato al servizio
  - `http://siteXXXX.tw.cs.unibo.it:8000/`
- Ricordarsi che la nostra configurazione di Docker prevede che la porta aperta per node.js sia la porta 8000!
- script è utile solo per node, ed è lo script javascript staticamente associato all'applicativo node, che fornisce le funzionalità attive.

I moduli di node

- Node è estremamente modulare. E' possibile mettere codice in file javascript ed eseguirlo secondo necessità.
- Il meccanismo di caricamento di moduli di node è semplice:
  - un modulo è un file Javascript.
  - Quando si richiede un modulo, esso viene cercato localmente o globalmente secondo un modello preciso.
- I moduli vengono caricati con `require()`.



npm

- I moduli di node vengono distribuiti ed installati con npm (*node package manager*)
- npm viene eseguito via command-line e interagisce con il registro npm.
  - meccanismo robusto per gestire dipendenze e versioni dei pacchetti (moduli)
  - semplice processo di pubblicazione di pacchetti e condividerli con altri utenti.
- In più: una piattaforma per repository pubbblici e privati, servizi enterprise (integrati con firewall aziendali), integrazione con altre piattaforme, ecc.

Usare npm

Un pacchetto npm è semplicemente un insieme di file, il più importante dei quali è *package.json*, che contiene metadati sul pacchetto (contenuto, dipendenze, ecc.)

Comandi utili di npm:

- npm init
  - Genera un file package.json globale nella directory in cui si trova. Tipicamente è la directory in cui eseguire node.
  - Viene anche creata la directory node_modules in cui verranno posizionati i package installati.
- npm install [package]
  - installa il pacchetto specificato, comprese tutte le dipendenze specificate, nella directory node_modules
- npm uninstall [package]
  - rimuove il pacchetto specificato, comprese tutte le dipendenze specificate e non usate da altri pacchetti installati
- npm update [package]
  - aggiorna esplicitamente il pacchetto specificato, comprese tutte le dipendenze specificate.

Express.js

- Express è un modello di server web per node.js
  - Open-source, licenza MIT
  - molto usato e molto supportato dalla comunità
  - molto semplice e molto espandibile con plugin
- Express si dedica al routing, alla gestione delle sessioni, all'autenticazione degli utenti, e molti altri principi fondanti delle connessioni HTTP.
- Nato nel 2010, acquistato nel 2015 da IBM, poi regalato da IBM alla Node.js Foundation.

Passport.js

- L'autenticazione può venire realizzata da package aggiuntivo di express.
  - Ce ne sono dozzine
  - Alcune autenticazioni base sono incluse direttamente in Express
  - Ad esempio *HTTP basic digest authentication*
- Passport.js è uno dei package più adottati
  - Flessibile e modulare
  - Supporta numerosissimi modelli di autenticazione: HTTP digest, Facebook, Google, Twitter, LinkedIn, ecc., chiamati *strategie*.
  - Permette anche di aggiungere nuove strategie ad-hoc.

Express/cors

Permette di definire server Express che realizzano completamente le funzionalità CORS (*Cross Origin Resource Sharing*).

- Per ragioni di sicurezza, un documento HTML può ricevere risorse (immagini, dati Ajax, script, ecc.) solo da un server posto nello stesso dominio del documento HTML di origine (*Same Origin Policy*).
- E' a discrezione del server permettere ad un'applicazione di un dominio diverso di richiedere le proprie risorse (*Cross Origin Resource Sharing*).
- Il browser, subito prima di eseguire un collegamento Ajax ad una risorsa di un dominio diverso, esegue un OPTION al server.
- Affinché sia poi possibile eseguire la richiesta, il server deve rispondere con alcune intestazioni HTTP specifiche, altrimenti il browser non effettuerà la richiesta vera e propria.
- Tipicamente: `Access-Control-Allow-Origin: *`

Il package express/cors inserisce automaticamente la risposta cors più appropriata e gestisce la creazione di servizi server-side aperti a tutti i richiedenti.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

MongoDB

MongoDB e il progetto

Il progetto di TW di quest'anno non impone in nessuna maniera l'uso di un database, né MongoDB né altri.

Tuttavia nel caso vogliate fare un po' di esperimenti con un DB noSQL, questa potrebbe essere una buona occasione.

In generale, un'applicazione web, anche data-oriented, non ha bisogno di un DB né SQL né di altro tipo, poiché tutti i linguaggi server-side (da Perl a PHP a node) forniscono metodi per utilizzare nativamente il file system e la applicazione può scrivere i suoi dati persistenti su un file qualunque.

Si usa un DB:

- quando il file diventa troppo grande per stare tutto in memoria,
- quando ho bisogno di metodi veloci per cercare un sottoinsieme di dati all'interno del file,
- quando ho bisogno di gestire accessi concorrenti in scrittura.

Usare un DB per leggere, *ma non modificare*, un file *intero* di dimensioni *non enormi* è come comprare una Ferrari per andare a far la spesa. Funziona, ma è complicato, ed è uno spreco di tempo e risorse.

MongoDB e il progetto

- Bisogna lanciare MongoDB su un docker separato ed autonomo.
- Usate gocker per lanciare MongoDB.
  - create mongoDB siteXXXX
- Mongo sarà accessibile all'indirizzo: mongodb://siteXXXX.tw.cs.unibo.it

MongoDB - Introduzione

MongoDB è un **database NoSQL orientato ai documenti.**
Esso memorizza i documenti in **JSON**, formato basato su JavaScript e più semplice di XML, ma comunque dotato di una buona espressività.

MongoDB - Introduzione

I documenti sono raggruppati in **collezioni** che possono essere anche eterogenee. Ciò significa che non c'è uno schema fisso per i documenti. Tra le collezioni non ci sono relazioni o legami garantiti da MongoDB.

MongoDB si adatta a molti contesti, in generale quando si manipolano grandi quantità di **dati eterogenei e senza uno schema**.

MongoDB - CRUD

Un **database**, è un insieme di collezioni che corrisponde ad un insieme di **file nel disco** fisso.

Un'istanza di MongoDB può gestire più database. Di seguito mostreremo brevemente come utilizzare la shell per effettuare le **operazioni CRUD** (**Create**, **Read**, **Update**, **Delete**).

MongoDB - CRUD

Per aprire la shell mongo eseguire il comando

```
mongo
```

Dopo aver aperto la shell, selezioniamo il database da usare:

```
use html_it
```

In realtà **non abbiamo ancora creato alcun database** con nome *html_it*.
Il DB *html_it* verrà **generato al momento del suo primo effettivo utilizzo**.

Tutti i comandi effettuabili via shell mongo possono essere effettuati anche attraverso javascript, come mostrato nelle prossime slide.

MongoDB - CRUD

Su MongoDB, non esiste alcun comando esplicito per creare un database.

Anche le collezioni vengono create implicitamente al primo utilizzo.

MongoDB - CRUD

Per **inserire** un documento, scrivi nella shell:

```
db.websites.insert(
    {
        name: "homepage",
        url: "http://www.html.it",
        tags: ["Development", "Design", "System"]
    }
);
```

La **risposta del server** sarà la seguente:

```
WriteResult({ "nInserted" : 1 })
```

e dice che è stato inserito un documento nel database.

MongoDB - CRUD

Per **interrogare tutti i documenti presenti nella collezione**, digitiamo quanto segue:

```
db.website.find()
```

..in questo caso il server risponderà così:

```
{
  "_id" : ObjectId("5450e468efaef47248f4412c"),
  "name" : "homepage",
  "url" : "http://www.html.it",
  "tags" : [ "Development", "Design", "System" ]
}
```

MongoDB - CRUD

Il campo _id **identifica univocamente ogni documento ed è obbligatorio**, oltre ad essere unico nell'intera collezione. Se non assegniamo noi un valore a _id (come abbiamo fatto in questo caso), MongoDB genera un **ObjectId** e lo assegna. Un ObjectId è un numero pseudocasuale che ha un tasso di collisione infinitesimo (è cioè molto difficile generarne due uguali).

Si noti anche che la sintassi di ObjectId non rientra nello standard JSON. Ciò perché in realtà MongoDB usa una sua estensione, chiamata **BSON** (**B**inary J**SON**).

MongoDB - CRUD

L’aggiornamento dei documenti avviene tramite il metodo **update**.

Vediamo subito un esempio:

```
db.websites.update(
    { name: "homepage"},
    { $set : { clicks: 5403 } },
    {multi: true}
)
```

MongoDB - CRUD

L’eliminazione di un documento dalla collezione viene effettuata per mezzo del metodo **remove**.

Il primo parametro rappresenta il criterio di selezione dei documenti da eliminare:

```
db.websites.remove( { name: "homepage"})
```

MongoDB - Server

MongoDB fornisce un driver per JavaScript **lato server** ossia per **Node.js**.

Per installarlo si usa il packet manager di Node.js, **npm**:

```
npm install mongodb
```

L'API di questo driver MongoDB è **event-driven**.

MongoDB - Server

Ad esempio, per accedere ad una collection su un database di MongoDB la sintassi è la seguente:

```
var client = require('mongodb').MongoClient;
client.connect("mongodb://localhost:27017/libri",
      function(error, libri) {
            if(! error) {
                  var autori = libri.collection("autori");
                autori.insert({ nome: "Umberto", cognome: "Eco" },
                  function(errorIns, risultatoIns) {  });
                db.close();
            }
      }
);
```

Sostituite mongodb://localhost:27017 con mongodb://siteXXXX.tw.cs.unibo.it

MongoDB - Client

Per quanto riguarda l'utilizzo di JavaScript **lato client**, non esiste un driver. Se si ha l'esigenza di interfacciarsi con JavaScript da una pagina web a MongoDB, bisogna passare per un servizio **REST**.

L'approccio quindi è di installare **un'interfaccia REST tra quelle disponibili**, in base alle proprie preferenze di framework, di piattaforma o di caratteristiche, quindi effettuare una chiamata AJAX al servizio REST.

MongoDB - Conclusioni

La guida in queste slide è stata presa da:

https://www.html.it/guide/guida-mongodb/

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

MongoDB - Installazione in locale, sulla VOSTRA macchina

MongoDB - Download

MongoDB è **open source.**

Nel **sito ufficiale** troviamo la disponibilità dei pacchetti di installazione per i principali sistema operativi: **Windows**, **Linux**, **Solaris** e **Mac OS X**.

MongoDB - Configurazione

Per **Linux**:

- Si consiglia di installare mongodb-org, cioè tutti i pacchetti.
- All'installazione viene creato l'utente mongodb con i privilegi per scrivere i propri files in /var/lib/mongodb e i log in /var/log/mongodb. Se si vogliono modificare queste impostazioni bisogna modificare il file /etc/mongodb.conf .

MongoDB - Configurazione

Linux: Per configurare MongoDb bisogna agire sul file **/etc/mongodb.conf**.

Principali impostazioni presenti nel file:

- **systemLog.path**: Percorso dei file di log
- **systemLog.destination**: Tipo di destinazione per il log. Indicare <code>syslog</code> o <code>file</code>
- **net.port**: Porta utilizzata dal server
- **storage.dbPath**: Percorso dei file contenenti il database

MongoDB - Configurazione

- **storage.directoryPerDB**: Booleano che indica se usare una directory per ogni DB. Abilitando questa impostazione è possibile fare in modo che MongoDB scriva e legga da diversi dischi rigidi contemporaneamente migliorando le performance globali e la quantità di spazio disponibile.

Su **Ubuntu**, se abbiamo lasciato tutte le impostazioni e i permessi sui file di default, basterà eseguire:

```
sudo service mongod start
```

MongoDB - Errori comuni

Il server va in **crash** e rimane bloccato! Descrizione del problema con esempio (e soluzioni): https://stackoverflow.com/questions/13700261/mongodb-wont-start-after-server-crash

Brevemente, come risolvere?

Segui la **guida** di MongoDB: Durability and Repair

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fabio Vitali

Dipartimento di Informatica – Scienze e Ingegneria
Alma mater – Università di Bologna

Fabio.vitali@unibo.it

www.unibo.it